A Real-Time Fall Detection and Alert System using a FSM on the DE10-Lite

Introduction

Falls represent a significant health hazard and are recognized as a leading cause of fatal and non-fatal injuries, particularly among the elderly population and other at-risk individuals. Research indicates that the medical severity of such injuries is frequently exacerbated not just by the impact itself, but by the subsequent delay in receiving timely medical assistance. To mitigate this risk, there is a growing need for reliable, automated systems capable of detecting accidents as they occur.

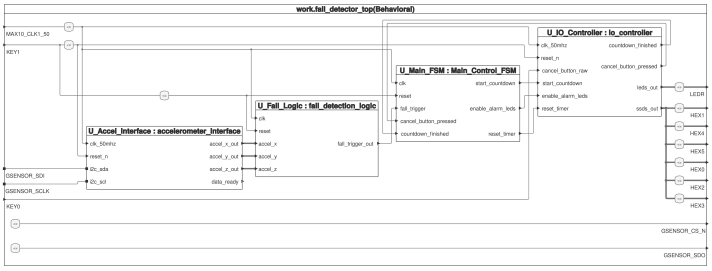
This project addresses this socially relevant problem by proposing the design of an automated, wearable monitoring device. Implemented as a synchronous sequential circuit on the DE10-Lite FPGA board, the system utilizes an onboard accelerometer to continuously monitor user movement. By distinguishing specific kinematic signatures—specifically a period of free-fall followed immediately by a high-impact shock—the device aims to provide a robust, real-time alert mechanism that ensures immediate attention following a fall event.

Objective

The objective of this project is to design, implement, and demonstrate a synchronous sequential circuit on the DE10-Lite board that functions as a real-time fall detector. The system uses the onboard accelerometer to detect a fall sequence, initiates a timed countdown on the 7-segment displays (SSDs) to allow the user to cancel a false alarm, and triggers a clear visual alert using the LEDs if the alarm is not canceled.

System Overview

The system is designed around a central Finite State Machine (FSM) and is modularized into four distinct functional blocks. The top-level entity (`fall_detector_top`) integrates these modules, connecting the 50MHz clock, hardware keys, and accelerometer pins to the internal logic.



figureTop-Level Block Diagram

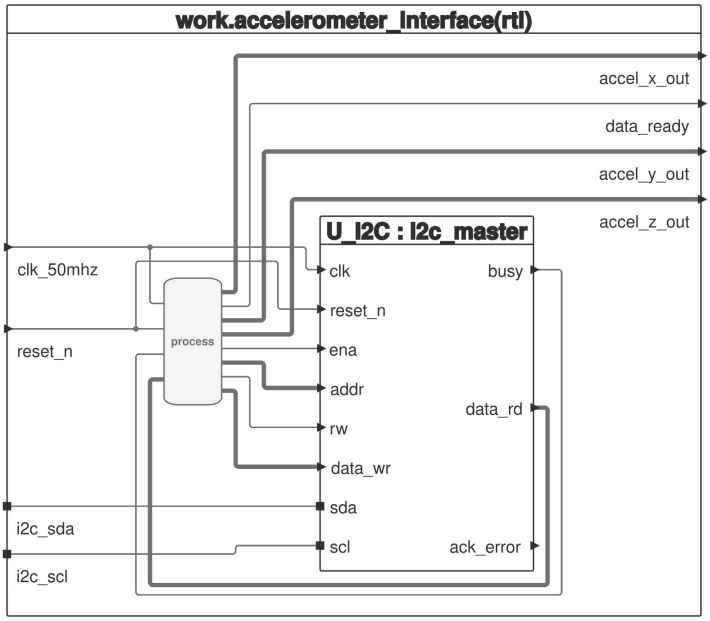
Implementation Details

The system was implemented entirely in VHDL using a modular design approach. The code was developed and linted using the **Sigasi Visual HDL** extension for VS Code to ensure syntax correctness before synthesis. The implementation is centered around four primary components:

1. Accelerometer Interface

The DE10-Lite board houses an ADXL345 digital accelerometer. To interface with this chip, we designed a custom **I2C Master** component. The interface logic operates at a bus speed of 100 kHz to ensure signal integrity.

Upon system reset, the interface executes a configuration state machine. It first writes to the `POWER_CTL` register (`0x2D`) to wake the device into measurement mode. It then sets the read pointer to `DATAX0` (`0x32`) and performs a continuous burst read of 6 bytes. This retrieves the X, Y, and Z low and high bytes in a single transaction, ensuring the data is synchronized. The 8-bit registers are then concatenated into three 16-bit signed integers for downstream processing.



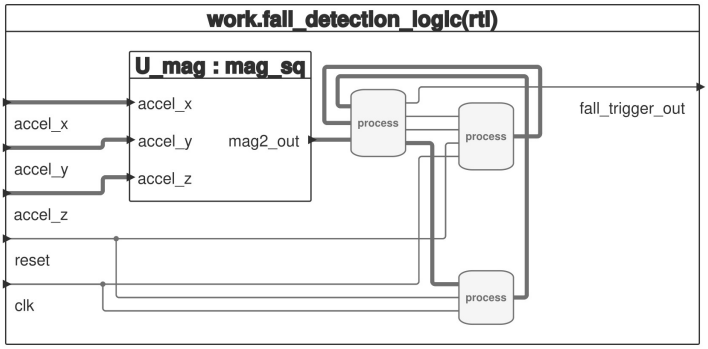
figureAccelerometer Interface

2. Fall Detection Logic

The core algorithmic challenge was to distinguish a fall from normal motion. We employed a magnitude-squared approach ($A^2 = X^2 + Y^2 + Z^2$) to avoid the high resource cost and latency of square-root calculations on the FPGA.

The detection logic utilizes a state machine with a timeout mechanism:

- **S_STARTUP:** The system waits here initially until the sensor stabilizes and reports a gravity magnitude consistent with being at rest. This prevents false triggers during the power-on sequence.
- **S_MONITOR:** The system compares the current A^2 against a `FREE_FALL_THRESHOLD` (experimentally set to 134,217,728). If the magnitude drops below this value, indicating weightlessness, the state transitions to `S_FREEFALL`.
- **S_FREEFALL:** A counter is started. If an acceleration spike exceeding the `IMPACT_THRESHOLD` (set to 1,254,456,536) occurs within approximately 1 second, a fall is confirmed, and the `fall_trigger_out` signal is asserted. If the timer expires without an impact, the system returns to monitoring.



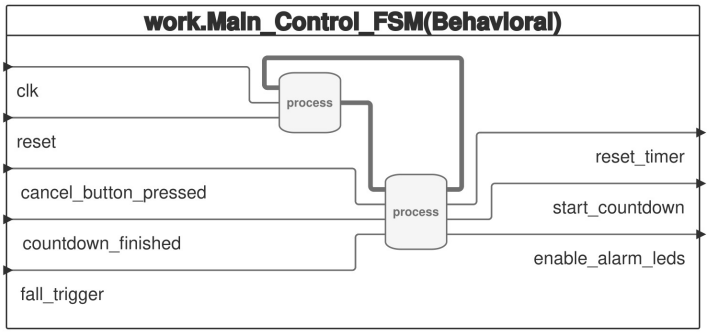
figureFall Detection Logic

3. Main Control FSM

The Main Control FSM acts as the system supervisor, coordinating the sensor inputs and the user interface. It is designed to be fail-safe, meaning the alarm state is the default destination after a trigger unless explicit human intervention occurs.

The state transitions are as follows:

- **STATE_MONITORING:** The system is idle. The `reset_timer` signal is held high to keep the countdown logic ready.
- **STATE_COUNTDOWN:** Upon receiving a high signal from the Fall Detection Logic, the FSM enters this state and asserts `start_countdown`. It waits for the `countdown_finished` signal from the I/O controller.
- **STATE_ALARM:** If the countdown completes, the FSM latches into this state, enabling the `enable_alarm_leds` signal to drive the strobe lights.
- **STATE_RESET:** If `KEY0` is pressed at any point during the countdown or alarm, the system transitions here to clear all flags before returning to monitoring.



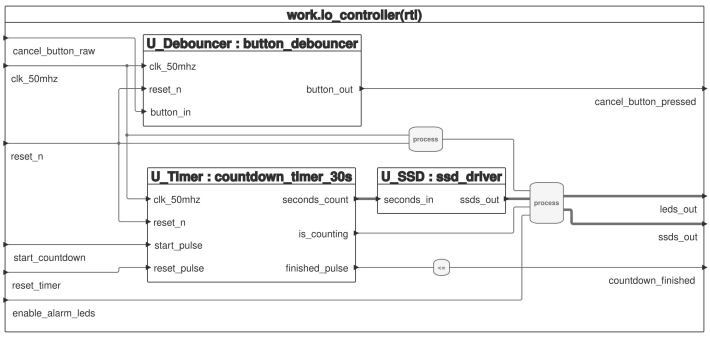
figureMain Control FSM

4. I/O Controller

The I/O Controller handles the physical interaction with the user. It includes a **Button Debouncer** which filters the raw input from the mechanical switches using a 20ms stability counter, ensuring that a single press is not interpreted as multiple glitches.

The visual feedback logic is multiplexed based on the system state. A custom `ssd_driver` converts the integer timer value into Binary Coded Decimal (BCD) for the 7-segment display.

- During **Monitoring**, the SSDs display "IDLE" using hardcoded segment patterns.
- During **Countdown**, the SSDs show the remaining seconds (30 down to 0).
- During **Alarm**, the logic toggles the SSDs to spell "HELP" and flashes the LEDs at 1Hz. This utilizes a clock divider to generate the visible strobe effect, maximizing visibility in an emergency.



figureI/O Controller

Results

The complete design was synthesized and programmed onto the MAX10 FPGA. The resource utilization was efficient, leaving ample room for future features such as VGA output or audio buzzers.

Experimental Verification: We conducted a series of physical tests to validate the thresholds chosen for the detection logic.

1. **Free-Fall Test:** The board was dropped from a height of 30cm onto a cushioned surface. The `mag_sq` logic

correctly computed the drop in acceleration (approaching zero), followed by the sharp spike upon landing. The system immediately transitioned to the countdown phase.

2. **False Positive Test:** The board was rotated and shaken moderately. The squared-magnitude calculation provided rotation-invariant data, and the thresholds were high enough that normal handling did not trigger the alarm.

3. **User Interface Test:** We verified the countdown timer against a stopwatch, confirming the 50MHz clock division logic was accurate. The "Cancel" button responded instantly, resetting the system to "IDLE".

4. **Alarm State:** Letting the timer expire resulted in the expected 1Hz visual strobe, confirming the emergency alert logic is functional.

Optimization and Challenges

- **I2C Clock Stretching:** Early iterations of the I2C Master failed to communicate with the ADXL345. Debugging revealed that the FSM was waiting indefinitely for an acknowledgment. We resolved this by enforcing a strict timing constraint and ignoring the slave's clock stretch, which is not required for this specific sensor at 100kHz.

- **Signal Noise:** The raw data from the MEMS sensor fluctuates even when stationary. We implemented a "Startup" state in the logic to discard the first few samples after reset, ensuring the calibration baseline was accurate.

Conclusion

This project successfully demonstrated a hardware-based solution for fall detection. By implementing the logic in VHDL on an FPGA, we achieved deterministic, real-time performance that is superior to software-based microcontroller polling. The modular design allows for easy expansion, and the current prototype meets all safety and functional objectives.

References

[1] Analog Devices, "ADXL345 Digital Accelerometer Datasheet," Rev. E, 2015.

[2] Terasic Technologies, "DE10-Lite User Manual," Version 2.1, 2017.

[3] Scott Larson, "I2C Master VHDL Logic," *DigiKey TechForum*, 2012.

[4] Sigasi, "Sigasi Visual HDL Extension for VS Code," *Visual Studio Marketplace*, 2024.

[5] Google, "Gemini AI," utilized for VHDL syntax consultation and error checking, 2025.

[6] Anthropic, "Claude AI," utilized for report structuring and optimization strategies, 2025.